REMARKS

These remarks are responsive to the Office Action mailed May 2, 2005 (hereinafter referred to as the "Office Action"). At the time of the last examination, Claims 1-4 and 7-27 were pending, of which claim 1 is an independent system claim, claim 4 is an independent method claim, with a corresponding independent computer program product claim 15, and claim 24 is an independent method claim. As shown above, all of the independent claims 1, 4, 15, and 24 (amongst some dependent claims) have been amended by this paper. Claims 8 – 9 and 18 – 19 are cancelled by this paper. Accordingly, upon entry of the amendments herein, Claims 1-4, 7, 10-17 and 20-27 will be pending for further consideration.

The Office Action rejected each of the pending independent claims (1, 4, 15, and 24) and all dependent claims, except claim 3, under 35 U.S.C. § 102(b) as being anticipated by TETware release 3.3 ("TETware") as described in the TETware User Guide, Revision 1.2 ("TETware UG") and the TETware Programmers Guide, Revision 1.2 ("TETware PG"). The Office Action rejected claim 3 under 35 U.S.C. § 103(a) as being unpatentable over TETware in view of U.S. Patent No. 6,505,342 to Hartmann et al. ("*Hartmann*").[1]

Applicants' invention, as recited for example in independent system claim 1, relates to a computer system for selecting and organizing individual test cases for use in testing a computer program to ensure that the program processes as intended. The system includes one or more program modules storing one or more available test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface; a harness client comprising a set of instructions that (i) receives user input specifying one or more filenames corresponding to the one or more program modules, (ii) executes a connector to scan for and discover the one or more available test cases that are stored in the one or more program modules and to organize the one or more available test cases into a test case hierarchy, and (iii) receives user input indicating which of the one or more available test cases in the test case hierarchy are selected to be executed on the computer program; a harness comprising a set of instructions that (i) receives the test case hierarchy, (ii) traverses the test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test

[1]Although the prior art status of the cited art is not being challenged at this time, Applicants reserve the right to do so in the future. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status or asserted teachings of the cited art.

suites comprise a test module, and (iii) executes each of the one or more available test cases that is selected to be executed on the computer program using the corresponding language and format independent interface of the selected test case to ensure that the computer program processes as intended; a connector comprising a set of instructions that (i) scans for the one or more available test cases stored in the one or more program modules, (ii) organizes the one or more available test cases into the test case hierarchy by extracting the one or more available test cases from the one or more program modules, and (iii) selectively integrates an interface between the test case hierarchy and the harness regardless of the language or format in which the one or more available test cases were written; and a processor for executing each selected test case, the harness, the harness client, and the connector.

Applicants' invention, as recited for example in independent method claim 4, relates to testing a computer program to determine whether the computer program processes as intended. The method includes a harness client (i) receiving user input that specifies one or more filenames to identify the program module, (ii) executes the connector to scan for and discover the one or more test cases of interest that are stored in the program module and to organize the one or more test cases of interest into a test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module, and (iii) receives user input indicating that the one or more test cases of interest in the test case hierarchy are to be executed on the computer program; a connector scanning the one or more test cases of interest stored in the program module, each test case having a language and format independent interface for executing the test case on the computer program regardless of the language or format used to develop the test case; the connector extracting the one or more test cases of interest from the program module; the connector organizing one or more test cases of interest into the test case hierarchy; the connector interfacing the harness with the one or more test cases of interest, wherein the interfacing allows the harness to recognize and execute the one or more test cases of interest regardless of the language or format in which the one or more test cases of interest were developed; and a harness traversing the test case hierarchy and executing each of the one or more test cases of interest to test the computer program. Independent claim 15 recites similar limitations from the perspective of a computer program product.

Applicants' invention, as recited for example in independent method claim 24, similarly relates to testing a computer program to determine whether the computer program processes as

intended. The method includes specifying one or more filenames for identifying one or more program modules storing one or more test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface; identifying the one or more test cases within the one or more program modules; translating the identified one or more test cases into a test case hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module; indicating that the one or more test cases in the test case hierarchy are to be executed on the computer program; providing an interface to the test case hierarchy in order to recognize and execute the one or more test cases regardless of the language or format in which the one or more test cases were written; and running each of the one or more test cases in the test case hierarchy to test the computer program.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." MPEP § 2131. That is, "for anticipation under 35 U.S.C. 102, the reference must teach every aspect of the claimed invention either explicitly or impliedly." MPEP § 706.02. Applicants also note that "[i]n determining that quantum of prior art disclosure which is necessary to declare an applicant's invention 'not novel' or 'anticipated' within section 102, the stated test is whether a reference contains an 'enabling disclosure.'" MPEP § 2121.01. In other words, a cited reference must be enabled with respect to each claim limitation. During examination, the pending claims are given their broadest reasonable interpretation, *i.e.*, they are interpreted as broadly as their terms reasonably allow, consistent with the specification. MPEP §§ 2111 & 2111.01.

As noted in Applicants' prior response, TETware discloses grouping test cases within a test suite. TETware PG, section 2.2. Test suites are organized as directory hierarchies—the top of each test suite directory is known as the test suite root directory. TETware UG, section 5.2.6. All files in a test suite reside below the test suite directory (or in a specified alternate execution directory). TETware PG, section 2.3; TETware UG, section 5.2.6. Test suites are required to include certain files and utilities, such as a build tool (e.g., make), a clean tool (e.g., rm), at least one test scenario file, etc. TETware PG, section 2.5.

A test scenario is a list of invocable components from a test suite that are processed during a particular TETware invocation. TETware UG, sections 2.2 & 5.3.2. Within a scenario file, a test case name may appear by itself or be attached to a directive that describes how the test

case should be executed (sequentially, in parallel with other test cases, repetitively, remotely, etc.). TETware PG, section 4.2.4.3; TETware UG, section 5.3.2.2. Test case names are interpreted relative to the test suite root directory or alternate execution directory, depending on the mode of operation. TETware UG, section 5.3.2.4. Section 5.3.2.5 of the TETware UG and section 4.4 of the TETware PG present some simple examples of test scenarios. Example test cases names listed in these scenarios follow the directory convention explained above (e.g. "/tset/tc1" and "/ts/tc1").

The test cases in a test suite are processed by a Test Case Controller (TCC), based on a chosen mode of operation (e.g., build, execute, clean up). TETware PG, section 2.5. In build mode, the TCC translates source test cases into executables, in execute mode the TCC loads and executes test cases, and in clean mode the TCC removes unwanted files. TETware PG, section 3.2. Each test case is an executable program. TETware PG, section 2.2. When a test case uses one of the TETware language specific APIs, its execution is supervised by a Test Case Manager (TCM). TETware PG, section 2.4.2. The TCM is not a separate program, but instead is linked with user-supplied test code and the API library to produce an executable test case. *Id.* There is a separate TCM module for each API that is supported by TETware. *Id.* For example, TETware includes a C TCM and a C++ TCM. TETware PG, section 2.4. Test cases are written to a specific language binding (C, C++, Shell, Korn Shell, etc.). TETware PG, sections 8, 9, 10, and 11.

Among other things, however, and in connection with the other recited claim limitations, TETware fails to teach, suggest, or enable a "test hierarchy in which one or more test cases comprise a test suite, and in which one or more test suites comprise a test module" as now recited in each of the independent claims. In contrast, TETware UG explicitly teaches away from this concept by stating that even though "[a] test suite is made up of one or more **test cases**" (TETware UG, section 2.1, second paragraph", "[a] **test suite** is the largest grouping of tests that can be processed by the TETware Test Case Controller" (TETware UG, section 2.1, first paragraph). Thus, TETware UG expressly teaches aware from a test hierarchy in which one or more test suites comprise a test module.

A similar feature was recited in the now cancelled dependent Claim 9, which also stood rejected under 35 U.S.C. 102(b) as being anticipated by TETware UG and TETware PG. However, that rejection was based on a false conclusion. For instance, the Office Action seems

to improperly equate the recited test module with the "scenario" described in section 2.2 of TETware UG. TETware UG and TETware PG nowhere describe that one or more test suites comprise a scenario. Instead, TETware UG states that a "**test scenario** is a list of one or more invocable components from a test suite". Even if the test scenario included all of the invocable components from a test suite, it still cannot be said that the test scenario **includes** the test suite, since there may be much more than invocable components in the test suite (see, for example, section 2.5 of TETware PG, which describes the structure of the test suite. This reason alone is sufficient to warrant a withdrawal of the 35 U.S.C. 102(b) reject of the independent claims. However, as will now be described, there are additional reasons why the independent claims are not anticipated by TETware.

As stated in the previous response to the Office Action, and in connection with the other recited claim limitations, TETware fails to teach, suggest, or enable the test case discovery aspect of Applicants' invention which is recited in each of the independent claims. For independent claims 1, 4, and 15, this discovery relates to receiving user input that specifies one or more filenames corresponding to one or more program modules, executing a connector to scan for and discover one or more available test cases that are stored in the one or more program modules and to organize the one or more available test cases into a test case hierarchy, and receiving user input that indicates which of the one or more available test cases in the test case hierarchy are selected to be executed on the computer program being tested. For independent claim 24, this discovery relates to specifying one or more filenames for identifying one or more one or more program modules storing one or more test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface, and identifying the one or more test cases within the one or more program modules.

In response to these arguments previously made, the Office Action states that these features are described in section 5.3.2 of TETware UG, without referring to specific language in that section to support that assertion. After reviewing section 5.3.2, it appears that the section just describes the scenario file, which seems to include scenario directives. However, TETware does not described that these directives causes the recited test case discovery functionality now recited in the claims. The undersigned respectfully requests clarification regarding what directives are considered to perform which functionality recited in the claims.

Accordingly, Applicants respectfully submit that the rejection of the independent claims under 35 U.S.C. § 102(b) as being anticipated by TETware has been overcome and should be withdrawn. As a result, the rejections of record with respect to the dependent claims also have been overcome and similarly should be withdrawn. *Hartmann* also fails to teach or suggest any of the recited features shown above to be lacking in the description of TETware. Accordingly, the 35 U.S.C. § 103(a) rejection should be withdrawn as well.

Based on at least the foregoing reasons, Applicants respectfully submit that the cited prior art fails to anticipate or make obvious Applicants invention, as claimed for example, in independent claims 1, 4, 15, and 24. Applicants note for the record that the remarks above render the remaining rejections of record for the independent and dependent claims moot, and thus addressing individual rejections or assertion with respect to the teachings of the cited art is unnecessary at the present time, but may be undertaken in the future if necessary or desirable, and Applicants reserve the right to do so.

In the event that the Examiner finds any remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney.

Dated this 13th day of July, 2005.

Respectfully submitted,

ADRIAN J. LEE
Registration No. 42,785
Attorney for Applicant
Customer No. 47973

AJL:ds
AJL0000000842V001